Introduction to Apigility

Rob Allen
March 2014

Nineteen Feet

Development : Training : Consultancy

http://19ft.com

APIs are becoming commonplace

APIs are hard

API considerations

- Content negotiation
- HTTP method negotiation
- Error reporting
- Versioning
- Discovery

Other considerations

- Validation
- Authentication
- Authorisation
- Documentation

An opinionated API builder

JSON

Hypermedia Application Language (HAL) - application/hal+json

```
{
    "_links": {
        "self": {
            "href": "http://localhost:8080/albums/1"
        }
    },
    "artist": "Eninem",
    "id": "1",
    "title": "The Marshall Mathers LP 2"
}
```

Error Reporting

API Problem - application/problem+json

```
{
    "type": "/api/problems/forbidden",
    "title": "Forbidden",
    "detail": "Your API key is missing or invalid.",
    "status": 403,
    "authenticationUrl": "/api/oauth"
}
```

HTTP Method Negotiation

```
POST /albums HTTP/1.1
Content-Type: application/json


405 Method Not Allowed
Allow: GET
```

OPTIONS

```
OPTIONS /albums HTTP/1.1
Content-Type: application/json


200 OK
Allow: GET
```

Accept

```
GET /albums/1 HTTP/1.1
Accept: application/xml

406 Not acceptable
Content-Type: application/problem+json

{
    "type": "/api/problems/content",
    "title": "Not acceptable",
    "detail": "This API can deliver
      application/vnd.music.v1+json, application/hal+json,
      or application/json only.",
    "status": 406
}
```

Content-Type

```
POST /albums HTTP/1.1
Content-Type: application/xml


415 Unsupported Media Type
Content-Type: application/problem+json

{
    "type": "/api/problems/content",
    "title": "Unsupported Media Type",
    "detail": "This API can accept
      application/vnd.music.v1+json, application/hal+json,
      or application/json only.",
    "status": 415
}
```

Versioning by default

Media type:

```
GET /albums HTTP/1.1
Accept: application/vnd.music.v1+json
```

URL-based:

```
/v1/albums
```

Validation

```
PATCH /albums/1 HTTP/1.1
Content-Type: application/json

{ "title": "" }


422 Unprocessable Entity
Content-Type: application/problem+json
{
  "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html",
  "title": "Unprocessable Entity",
  "detail": "Failed validation",
  "status": 422,
  "validation_messages": {
      "title": "Invalid title; must be a non-empty string"
  }
}
```

Authentication

- HTTP Basic and Digest (for internal APIs)
- OAuth2 (for public APIs)
- Event-driven, to accommodate anything else
- Return a problem response early if invalid credentials are provided

Authentication

```
GET /albums/1 HTTP/1.1
Authorisation: Basic foobar
Accept: application/json


401 Unauthorized
Content-Type: application/problem+json

{
  "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html",
  "title": "Unauthorized",
  "detail": "Unauthorized",
  "status": 401
}
```

Authorisation

```
GET /albums/1 HTTP/1.1
Accept: application/json


403 Forbidden
Content-Type: application/problem+json

{
    "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html",
    "title": "Forbidden",
    "detail": "Forbidden",
    "status": 403
}
```

Hyperlinking: Pagination

Automatic when you return `Zend\Paginator\Paginator`.

```
{
    _links: {
        self: { href: "/api/albums?page=3" },
        first: { href: "/api/albums" },
        last: { href: "/api/albums?page=14" },
        prev: { href: "/api/albums?page=2" },
        next: { href: "/api/albums?page=4" }
    }
}
```

Documentation

- Written within admin while setting up API
- Automatically populated via validation admin
- User documentation:
    - apigility/documentation/{API name}/V1
    - JSON or HTMl based on accept header
    - Swagger available too

Let’s see it in action then!

Getting Started

Install:

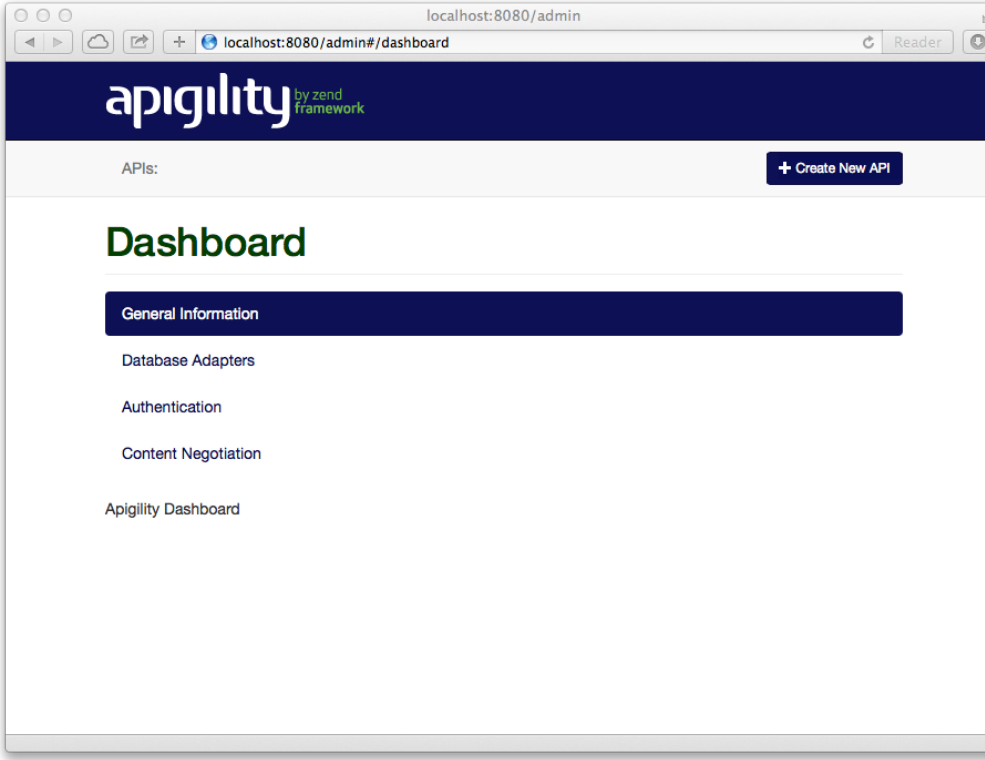
```
$ composer.phar create-project \
  -sdev zfcampus/zf-apigility-skeleton music
```

Development Mode:

```
$ cd music
$ php public/index.php development enable
```

Run the admin web UI:

```
$ php -S 0:8080 -t public/ public/index.php
```


localhost:8080/admin
localhost:8080/admin#/dashboard
Reader
apigility by zend framework
APIs:
+ Create New API
Dashboard
General Information
Database Adapters
Authentication
Content Negotiation
Apigility Dashboard

Create your API

(Screencast)

Test with curl

```
$ curl -s -H "Accept: application/vnd.music.v1+json" \
  http://localhost:8080/albums | python -mjson.tool

{
  "detail": "The GET method has not been defined for
             collections",
  "status": 405,
  "title": "Method Not Allowed",
  "type": "w3.org/Protocols/rfc2616/rfc2616-sec10.html"
}
```

Source code

Apigility has created a module called `Music` for our API.

The `Album` endpoint is in: `src/Music/V1/Rest/Album`

Classes:

`AlbumResource`	entry point to service
`AlbumCollection`	a collection of albums
`AlbumEntity`	a single album

AlbumResource

Collection of albums: `/albums`

Class method	HTTP	Notes
`fetchAll`	GET	retrieve all items
`create`	POST	create an item
`replaceList`	PUT	replace all items
`deleteList`	DELETE	Delete all items

AlbumResource

Single album: `/albums/[album_id]`

Class method	HTTP	Notes
`fetch`	GET	retrieve an item
`patch`	PATCH	update some fields
`update`	PUT	replace an item
`delete`	DELETE	delete an item

The data model

```
class AlbumEntity
{
    protected $id;
    protected $artist;
    protected $title;
}

class AlbumMapper
{
  public function fetchAll($filter) { /* .. */ }
  public function fetchOne($id) { /* .. */ }
  public function save($album) { /* .. */ }
}
```

Fetching the collection

```
class AlbumResource extends AbstractResourceListener
{
  public function fetchAll($params = array())
  {
    // return an AlbumCollection
    return $this->mapper->fetchAll($params);
  }
}
```

Test:

```
$ curl -s -H "Accept: application/vnd.music.v1+json" \
  http://localhost:8080/albums?title=night \
  | python -mjson.tool
```

Fetching the collection


```
{
    "_embedded": {
        "album": [
            {
                "_links": {
                    "self": { "href": "http://localhost:8080/albums/4" }
                },
                "artist": "Andre Rieu",
                "id": "4",
                "title": "Music of the Night"
            }
        ]
    },
    "_links": {
        "first": { "href": "http://localhost:8080/albums" },
        "last": { "href": "http://localhost:8080/albums?page=1" },
        "self": { "href": "http://localhost:8080/albums?page=1" }
    },
    "page_count": 1,
    "page_size": 25,
    "total_items": 1
}
```

Fetching a single resource

```
class AlbumResource extends AbstractResourceListener
{
  public function fetch($id)
  {
    // return an AlbumEntity
    return $this->mapper->fetchOne($id);
  }
}
```

Test:

```
$ curl -s -H "Accept: application/vnd.music.v1+json" \
  http://localhost:8080/albums/1 | python -mjson.tool
```

Fetching a single resource


```
{
    "_links": {
        "self": {
            "href": "http://localhost:8080/albums/1"
        }
    },
    "artist": "Eninem",
    "id": "1",
    "title": "The Marshall Mathers LP 2"
}
```

Creating a resource

```
public function create($data)
{
  // return an AlbumEntity
  return $this->mapper->save($data);
}
```

POST to the collection

```
$ curl -s -X POST -H "Content-type: application/json" \
  -H "Accept: application/vnd.music.v1+json"  \
  -d '{"title":"True", "artist":"Avicii"}' \
  http://localhost:8080/albums |  python -mjson.tool
```

Creating a resource

Header:

```
HTTP/1.1 201 Created
Location: http://localhost:8080/albums/7
```

Body:

```
{
    "_links": {
        "self": {
            "href": "http://localhost:8080/albums/7"
        }
    },
    "artist": "Avicii",
    "id": "7",
    "title": "True"
}
```

Updating a resource

```
public function update($id, $data)
{
  // return an AlbumEntity
  return $this->mapper->save($data, $id);
}
```

PUT to the resource

```
$ curl -s -X PUT -H "Content-type: application/json" \
  -H "Accept: application/vnd.music.v1+json"  \
  -d '{"title":"True!", "artist":"Avicii"}' \
  http://localhost:8080/albums/7 |  python -mjson.tool
```

Updating a resource

Header:

```
HTTP/1.1 200 OK
```

Body:

```
{
    "_links": {
        "self": {
            "href": "http://localhost:8080/albums/7"
        }
    },
    "artist": "Avicii",
    "id": "7",
    "title": "True!"
}
```

Validation

- Built into the Apigility admin
- Tested when routing: very fast to fail

Validation

(Screencast)

Validation

POST with an an empty artist to the collection

```
$ curl -s -X POST -H "Content-type: application/json" \
  -H "Accept: application/vnd.music.v1+json"  \
  -d '{"title":"Greatest Hits", "artist":""}' \
  http://localhost:8080/albums |  python -mjson.tool
```

Validation

Header:

```
HTTP/1.1 422 Unprocessable Entity
```

Body:

```
{
    "detail": "Failed Validation",
    "status": 422,
    "title": "Unprocessable Entity",
    "type": "http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html",
    "validation_messages": {
        "artist": {
            "isEmpty": "Value is required and can't be empty"
        }
    }
}
```

To sum up

Apigility provides the boring bits of API building:

- Content negotiation
- Discovery (HATEOS) via application/hal+json
- Error reporting via application/problem+json
- Versioning
- Validation
- Authentication
- Documentation

1.0 soon!

Thank you!

https://joind.in/10915

Rob Allen - http://akrabat.com - @akrabat